Prof. Dr. Christian Steger
Prof. Dr. Reinhard Wilhelm

Sebastian Altmeyer, M.Sc.
Michael Gerke, M.Sc.
Dipl.-Inf. Hans-Jörg Peter

Embedded Systems 2010/2011 – Milestone 2: Implementation

Due: Tuesday, 1st February 2011, *before* the lecture (i.e., 10:10)

Implementation

The second milestone is the actual implementation of RoboDog.

1. Download the template `Robot.tar` from the course website and unpack it.
2. Rename the directory to `RoboDogXY`, where `XY` is your group number (see schedule).
3. Generate the C-Code from your Scade-model.
4. Copy this code (e.g., the directory `KCG`) into the working directory (from step 2).
5. Adapt `Makefile` and `RoboDog.c`.
6. Connect the mindstorm to your computer.
7. Download the executable `RoboDog.rxe` to the mindstorm using `rxeflash.sh` (within `Cygwin/Eclipse`).
8. Test your implementation.

The last 3 steps are restricted to room 401 and to the scheduled time slot for each group.

Makefile

Make sure that the directory `KCG` is part of your general working directory. For each operator defined and used in your Scade-model, include the corresponding C-file in `Makefile`:

```
TARGET_SOURCES := \
        RoboDog.c \
        KCG/SCADEOPERATOR1.c \
        KCG/SCADEOPERATOR2.c
```

Note that you have to replace `SCADEOPERATORx` with the actual names of your operators.

RoboDog.c

First, you have to include the header files of your Scade model:

```
#include <KCG/SCADEOPERATOR.h>
```

The following C-Code is used to access the state-machine

```
outC_SCADEOPERATOR result;    // stores INPUT/OUTPUT of your state-machine
SCADEOPERATOR_reset(&result); // initializes the state-machine
SCADEOPERATOR(&result);       // computes the next cycle of your state-machine
```

The struct `outC_SCADEOPERATOR` is defined in the generated file `SCADEOPERATOR.h`. Note that you have to replace `SCADEOPERATOR` by the actual name of your defined operator.

Accessing Sensors/Actuators

The following functions are used to access the sensors/actuators:

```
  v = ecrobot_get_sonar_sensor(NXT_PORT_S1);
  x = ecrobot_get_light_sensor(NXT_PORT_S2);
  y = ecrobot_get_touch_sensor(NXT_PORT_S3);
  z = ecrobot_get_sound_sensor(NXT_PORT_S4);

  nxt_motor_set_speed(NXT_PORT_A, RightValue, 1);
  nxt_motor_set_speed(NXT_PORT_B, LeftValue, 1);
  where  v,x,y,z, RightValue, LeftValue have to be integers.
```

The sensor values either have to be global variables in case they have been defined as SENSORS in the Scade model or are part of the struct `outC_SCADEOPERATOR` in case they are defined as input to the Scade model.

The following example shows how to output values on the display:

```
display_clear(0);
display_goto_xy(0, 0); // show on first line
display_string("Hello World!");
display_goto_xy(0, 1); // show on second line
display_string("Value:" );
display_int(value, 0);
display_update();
```

On the mindstorm display, it shows

```
Hello World!
Value: 42
```

Timing, Constants, and Delays

The computation of the next cycle of your state machine (`SCADEOPERATOR(&result);`) has to be repeatedly called within the `while`-loop. You also have to use and adapt the command

```
systick_wait_ms(some_int_value);
```

to ensure the right timing of your model.

The generated file `kcg_consts.h` contains all constants defined within Scade. Adapt these values such that your model behaves correctly.

Submission

Add a file `group.txt` to this directory containing the name and matr. number of each group member. Compress the whole working directory (named `RoboDogXY`) either as a .rar or .zip archive comprising the directory `KCG` as well as the final executable `RoboDog.rxe` and send it to:

`altmeyer@cs.uni-saarland.de`

In addition, provide a print-out of the graphical representation of the model as well as a short explanation. Only submissions available on paper **and** via mail will be graded.